Networked Services

Loren McIntyre / mcint

Spring 22, OCF/XCF Linux SysAdmin DeCal

Advanced 7

Networked Services

Loren McIntyre / mcint
Fall 2020, OCF/XCF Linux SysAdmin DeCal

Who am I?

Platform Engineer (frmr) at Kloudless, Berkeley startup

Former OCF General Manager, Root Staff member

Depicted at the base of Half Dome!

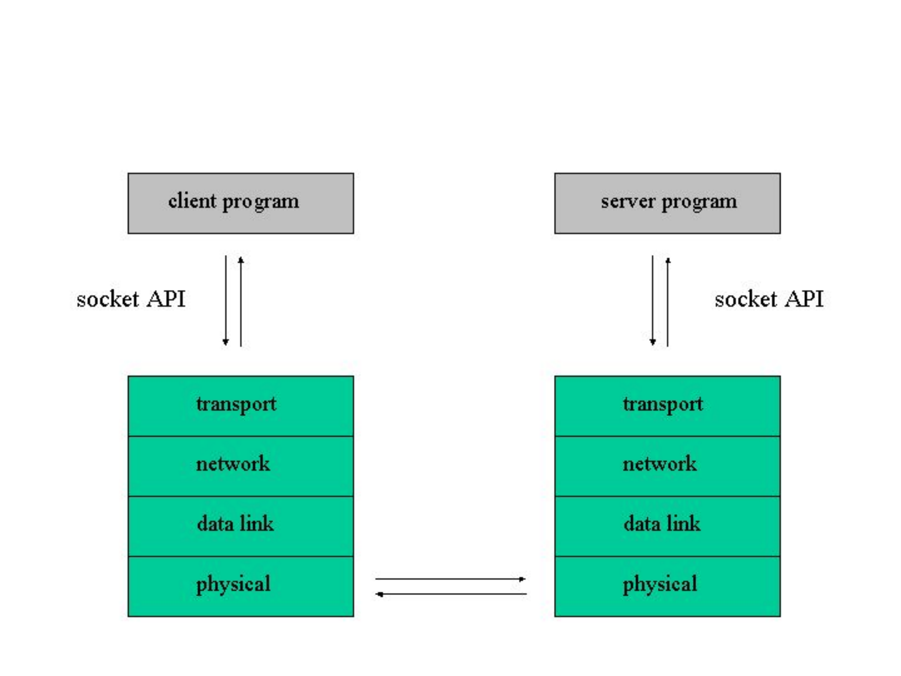
Slides: slightly modified from Fall 2019 presentation by Jason Perrin!

What’s special about networked services?

- Most services are networked
  - Metcalf’s Law: value of network ~ (# users)^2
- More security concerns
  - Malicious users, accidentally abusive users
  - Rate limiting, ACLs, firewalls, etc.
- Need to deal with clients
  - Load balancing, timeouts
- Centralize common services

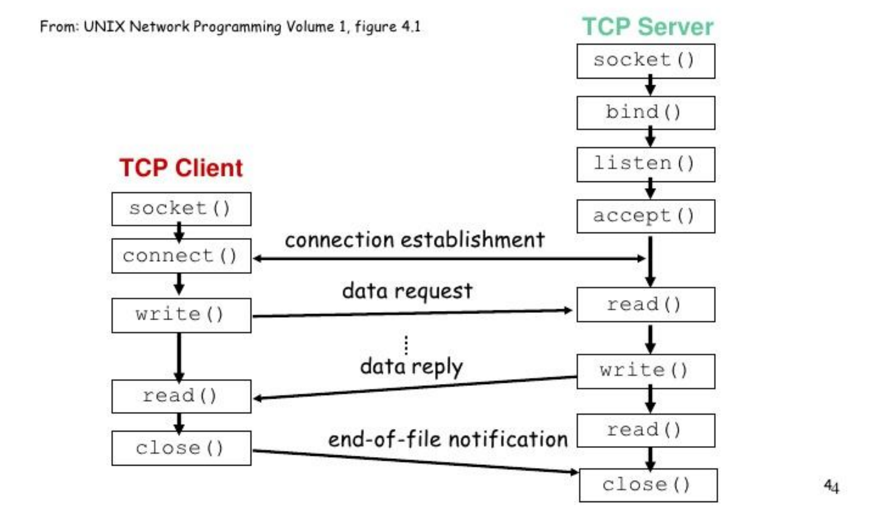
Sockets

- Mentioned in the networking lecture
- Very heavily used in unix
- Familiar interface with the network for programmers
- Many different types of sockets
  - For today: Internet sockets only



The socket connection process

- Listening program (service) creates a socket and listens for incoming connections
- Bunch of common functions given in C for doing this
- For more info:
  - CS 162 HW 2 (Spring 2019)
  - Beej's Guide to Network Programming



Popular networked service examples

A (relatively) small list of examples

- There are many more services than these, these are just some common ones (grouped):
  - DNS
  - NTP
  - SSH
  - NFS
  - LDAP
  - Kerberos
  - Web servers
  - Databases
  - Mail servers
  - Load balancers
- The list goes on and on and on...

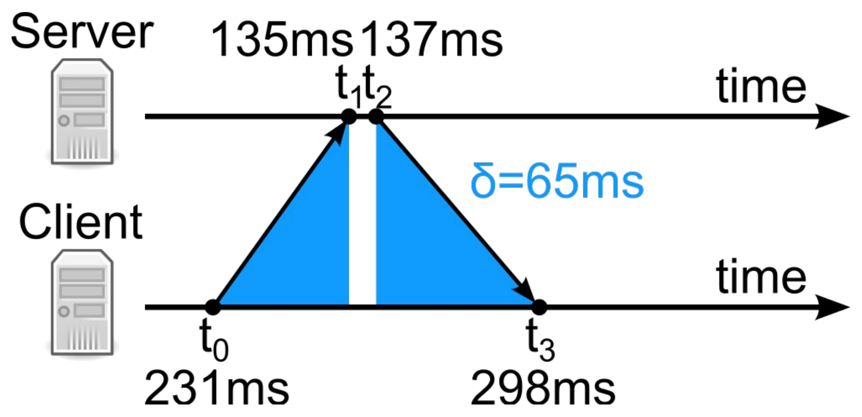
DNS: Domain Name System

- Recap: maps from something like "`ocf.berkeley.edu`" (a domain name) -> `169.229.226.23` (an IP)
  - Or to `2607:f140:8801::1:23` for IPv6
- Most common: BIND (Berkeley Internet Name Daemon), Route 53, NS1, etc.

THE RUMORS ARE TRUE. GOOGLE WILL BE SHUTTING DOWN PLUS— ALONG WITH HANGOUTS, PHOTOS, VOICE, DOCS, DRIVE, MAPS, GMAIL, CHROME, ANDROID, AND SEARCH— TO FOCUS ON OUR CORE PROJECT: THE 8.8.8.8 DNS SERVER.
Google

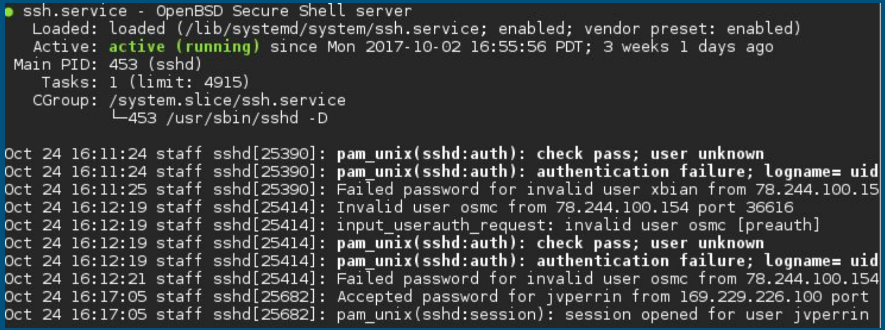
NTP: Network Time Protocol

- One of the oldest protocols still in current use (in use since before 1985)
- Wouldn’t typically think that a system’s clock could be a problem
- Some protocols need the clock to be reasonably accurate (verifying SSL certs, Kerberos, etc.)
- Critical for security, encryption, and many other tasks

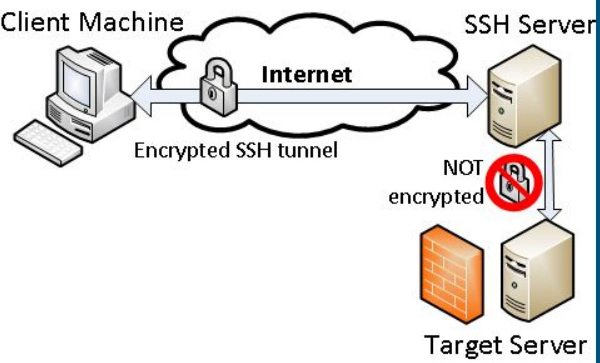


SSH: Secure Shell

- What you have all been using to access your VMs! (openssh)
- This is a service just as much as any other, it’s just particularly important to keep running because connecting and fixing stuff is much harder if it breaks.
- Actually can be used as a tunnel to encrypt other kinds of traffic, so for instance can browse the web by tunneling it over SSH

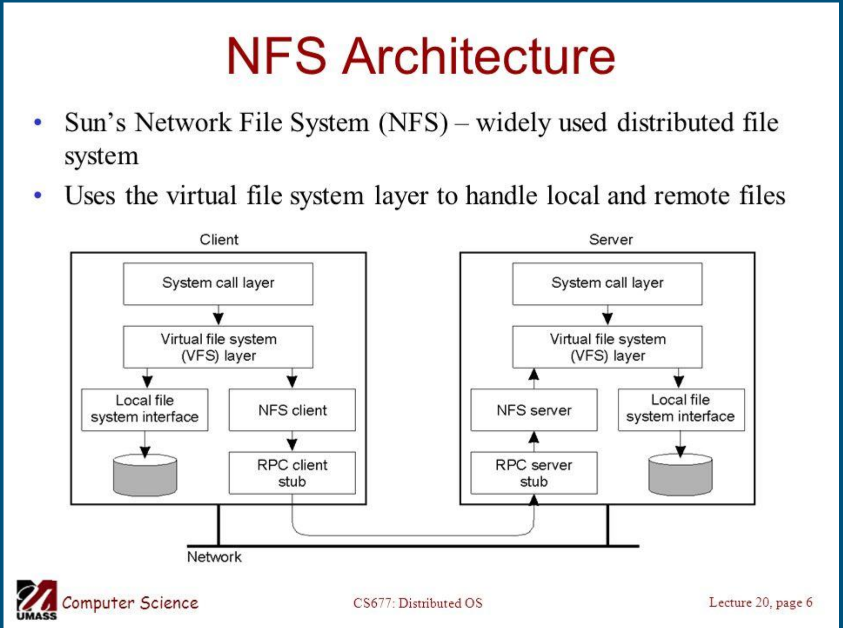


```
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2017-10-02 16:55:56 PDT; 3 weeks 1 days ago
 Main PID: 453 (sshd)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/ssh.service
           └─453 /usr/sbin/sshd -D

Oct 24 16:11:24 staff sshd[25390]: pam_unix(sshd:auth): check pass; user unknown
Oct 24 16:11:24 staff sshd[25390]: pam_unix(sshd:auth): authentication failure; logname= uid
Oct 24 16:11:25 staff sshd[25390]: Failed password for invalid user xbian from 78.244.100.15
Oct 24 16:12:19 staff sshd[25414]: Invalid user osmc from 78.244.100.154 port 36616
Oct 24 16:12:19 staff sshd[25414]: input_userauth_request: invalid user osmc [preauth]
Oct 24 16:12:19 staff sshd[25414]: pam_unix(sshd:auth): check pass; user unknown
Oct 24 16:12:19 staff sshd[25414]: pam_unix(sshd:auth): authentication failure; logname= uid
Oct 24 16:12:21 staff sshd[25414]: Failed password for invalid user osmc from 78.244.100.154
Oct 24 16:17:05 staff sshd[25682]: Accepted password for jvperrin from 169.229.226.100 port
Oct 24 16:17:05 staff sshd[25682]: pam_unix(sshd:session): session opened for user jvperrin
```



NFS: Network File System

- Used to share files between multiple servers, created in 1984
- Files live on a server, shared with clients
- Clients can treat files as if local
- This makes it easier, for instance, to edit files on one server, and have them run on another server
  - We do this for OCF web hosting
- Clients don’t need a lot of disk



NFS Architecture

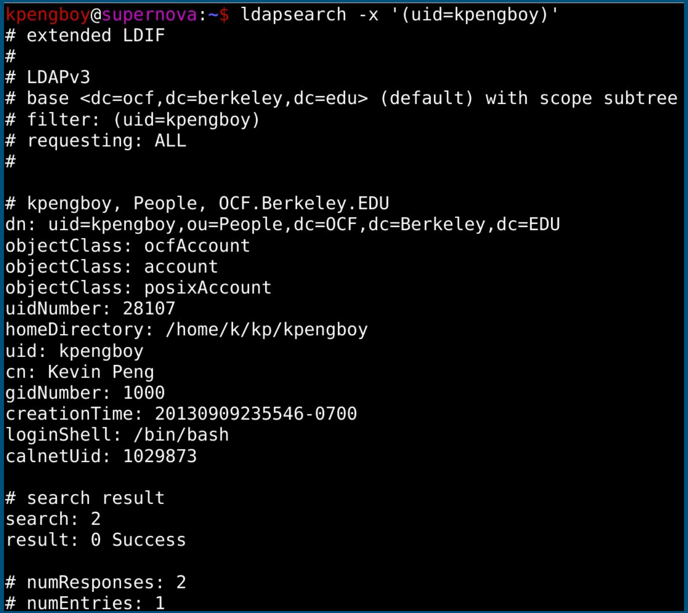
- Sun’s Network File System (NFS) – widely used distributed file system
- Uses the virtual file system layer to handle local and remote files

Client
System call layer
Virtual file system (VFS) layer
Local file system interface
NFS client
RPC client stub
Server
System call layer
Virtual file system (VFS) layer
NFS server
Local file system interface
RPC server stub
Network

UMASS Computer Science CS677: Distributed OS Lecture 20, page 6

LDAP: Lightweight Directory Access Protocol

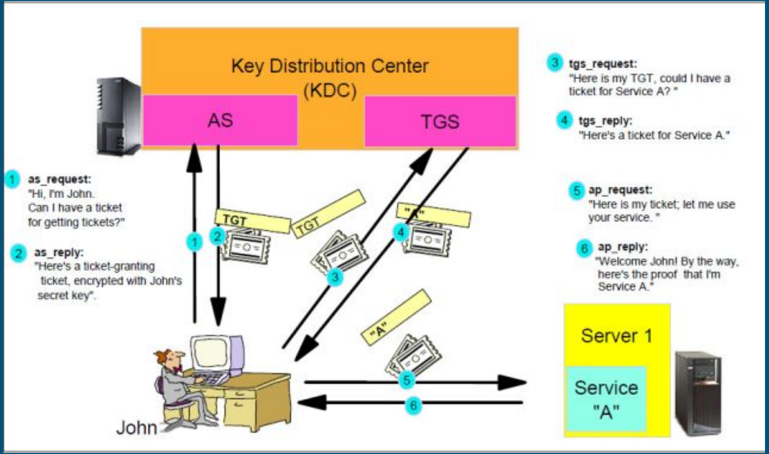
- A relatively simple directory service (like a phonebook) that stores data about users
- Organized hierarchically, as shown in the “dn” attribute (EDU, then Berkeley, then OCF, etc.)
- LDAP server stores this info, clients query it over network for authentication etc.
- Often used for authentication when supporting Windows (Active Directory)



```
kpengboy@supernova:~$ ldapsearch -x '(uid=kpengboy)'
# extended LDIF
#
# LDAPv3
# base <dc=ocf,dc=berkeley,dc=edu> (default) with scope subtree
# filter: (uid=kpengboy)
# requesting: ALL
#

# kpengboy, People, OCF.Berkeley.EDU
dn: uid=kpengboy,ou=People,dc=OCF,dc=Berkeley,dc=EDU
objectClass: ocfAccount
objectClass: account
objectClass: posixAccount
uidNumber: 28107
homeDirectory: /home/k/kp/kpengboy
uid: kpengboy
cn: Kevin Peng
gidNumber: 1000
creationTime: 20130909235546-0700
loginShell: /bin/bash
calnetUid: 1029873

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Kerberos

- Network authentication protocol based on tickets
    - Authenticate once, get a ticket, don’t have to type your password again!
    - Designed to work in the face of an insecure network
- Often used in Windows
- Quite useful on Unix too



1 **as_request:**
"Hi, I'm John.
Can I have a ticket
for getting tickets?"

2 **as_reply:**
"Here's a ticket-granting ticket, encrypted with John's secret key".

3 **tgs_request:**
"Here is my TGT, could I have a ticket for Service A? "

4 **tgs_reply:**
"Here's a ticket for Service A."

5 **ap_request:**
"Here is my ticket; let me use your service. "

6 **ap_reply:**
"Welcome John! By the way, here's the proof that I'm Service A."

Web Servers

- NGINX (pronounced engine-x)
  - Designed for concurrency, newer and generally faster than Apache, often used a proxy in front of other services
- Apache
  - Slower than nginx, but more established and generally has more features and modules available
- Plenty more (lighttpd, cherokee, etc.), but NGINX and Apache are the main two you will encounter over and over

(if you look up Apache, you'll find lots of these feathers from various years/designs)

Databases

- Databases can be accessible from the network too!
  - PostgreSQL and MySQL/MariaDB are the most common relational databases
- Very useful for any kind of dynamic web applications
  - Wordpress, Django, Ruby on Rails, etc.
- Could have a local database, but if you have a lot of applications, you'd need a lot of individual database applications

Mail Servers

- Receive mail and
  - Deliver it elsewhere
  - Or store it locally for you to fetch and read
- Many choices of software
  - Transfer: Postfix, Exim
  - Delivery/Storage: Dovecot
- Common protocols used: SMTP, POP3, IMAP

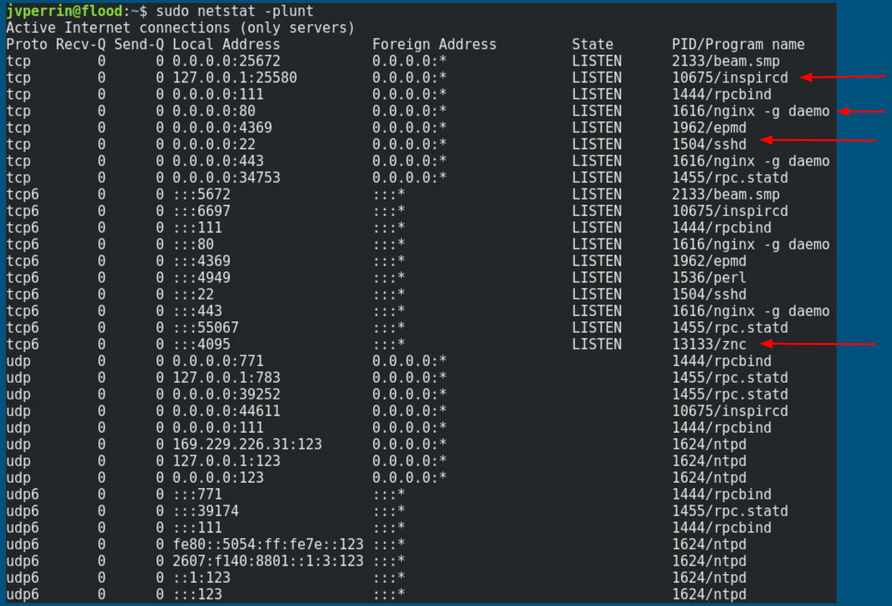
Load balancers

- Handling requests for a service all in one place doesn't scale since you can have millions of clients at a time
- HAProxy
  - Common open-source load balancer
  - Accepts connections, and then sends them on to somewhere else to be answered (typically forwards on to another server)
- NGINX is actually starting to do this too, along with being a web and proxy server
- Envoy (more of a service mesh provider, but we'll allow it)

HAPROXY
Powering Your Uptime

How do you know what networked services are running?

- `sudo netstat -l`, but you typically don’t care about a bunch of internal sockets
- More useful: `sudo netstat -plunt` or `sudo netstat -peanut`
- Can also use `ss` command



```
jvperrin@flood:~$ sudo netstat -plunt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:25672           0.0.0.0:*               LISTEN      2133/beam.smp
tcp        0      0 127.0.0.1:25580         0.0.0.0:*               LISTEN      10675/inspircd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      1444/rpcbind
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1616/nginx -g daemo
tcp        0      0 0.0.0.0:4369            0.0.0.0:*               LISTEN      1962/epmd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1504/sshd
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      1616/nginx -g daemo
tcp        0      0 0.0.0.0:34753           0.0.0.0:*               LISTEN      1455/rpc.statd
tcp6       0      0 :::5672                 :::*                    LISTEN      2133/beam.smp
tcp6       0      0 :::6697                 :::*                    LISTEN      10675/inspircd
tcp6       0      0 :::111                  :::*                    LISTEN      1444/rpcbind
tcp6       0      0 :::80                   :::*                    LISTEN      1616/nginx -g daemo
tcp6       0      0 :::4369                 :::*                    LISTEN      1962/epmd
tcp6       0      0 :::4949                 :::*                    LISTEN      1536/perl
tcp6       0      0 :::22                   :::*                    LISTEN      1504/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      1616/nginx -g daemo
tcp6       0      0 :::55067                :::*                    LISTEN      1455/rpc.statd
tcp6       0      0 :::4095                 :::*                    LISTEN      13133/znc
udp        0      0 0.0.0.0:771             0.0.0.0:*                           1444/rpcbind
udp        0      0 127.0.0.1:783           0.0.0.0:*                           1455/rpc.statd
udp        0      0 0.0.0.0:39252           0.0.0.0:*                           1455/rpc.statd
udp        0      0 0.0.0.0:44611           0.0.0.0:*                           10675/inspircd
udp        0      0 0.0.0.0:111             0.0.0.0:*                           1444/rpcbind
udp        0      0 169.229.226.31:123      0.0.0.0:*                           1624/ntpd
udp        0      0 127.0.0.1:123           0.0.0.0:*                           1624/ntpd
udp        0      0 0.0.0.0:123             0.0.0.0:*                           1624/ntpd
udp6       0      0 :::771                  :::*                                1444/rpcbind
udp6       0      0 :::39174                :::*                                1455/rpc.statd
udp6       0      0 :::111                  :::*                                1444/rpcbind
udp6       0      0 fe80::5054:ff:fe7e::123 :::*                                1624/ntpd
udp6       0      0 2607:f140:8801::1:3:123 :::*                                1624/ntpd
udp6       0      0 ::1:123                 :::*                                1624/ntpd
udp6       0      0 :::123                  :::*                                1624/ntpd
```

(This is running on an IRC server, there are a lot of other networked services running)

Any lingering questions?